Implementation

- Faithful translation of a design into a target environment
- Design should be free of target-environment dependencies
- Should generate target software algorithmically
  - This preserves the application model as the source and treats code as another view.
  - Changes should be made in the application model, not the code.
  - In the absence of tools to do this translation automatically, this ideal is hard to achieve.

ODMG
(Object Database Management Group)

- Goal: Develop a standard for object databases.
- Approach: Transparently integrate OO languages with DBs
  - make the query language and the programming language one and the same
  - extend OO languages with persistence, concurrency control, crash recovery, and query processing
  - extend DBs by making objects appear like programming objects in one of several OO programming languages
- Components
  - ODL: Object Definition Language
  - OML: Object Manipulation Language
  - OQL: the query part of OML
ODMG Language Bindings

- Examples: C++, Smalltalk, Java
- Provides a representation of ODMG ODL & OML in terms of the OO language
- L+ Program → Executable Code
  - L+ Program (L+ ODL, L OML, L Auxiliary Code)
  - Preprocessor: L+ ODL → DB Schemes & L Declarations
  - Linker: L Object Code, L DB Runtime Package (provided code library) → Executable Code (which runs the application and accesses the DB)

ODMG ODL

- **interface** – names an ODL declaration
- **extent** – names the set of objects declared
- **key[s]** – declares keys
- **persistent | transient** – makes the extent persistent or transient
- **attribute** – declares an attribute
- **readonly** – makes an attribute a read-only attribute
- **Set | List | Bag | Array** – declares a collection type
- **relationship** – declares a relationship
- **inverse** – declares an inverse relationship
B&B Example –
Generated Database Scheme

Room(RoomNr, RoomName, NrBeds, Cost)

Guest(GuestNr, GuestName, StreetNr, City)

Reservation(GuestNr, RoomNr, ArrivalDate, NrDays)

Room(RoomNr) $\rightarrow$ Reservation[RoomNr]
Guest[GuestNr] = Reservation[GuestNr]

B&B Example –
ODMG ODL Abstraction Diagram

Room(RoomNr, RoomName, NrBeds, Cost)

Reservation(GuestNr, RoomNr, ArrivalDate, NrDays)

Guest(GuestNr, GuestName, StreetNr, City)

is reserved in
is for
has
is for
B&B Example – ODL for Room

interface Room {
  extent Rooms
  keys RoomNr, RoomName
  : persistent {
    attribute Unsigned Short RoomNr;
    relationship Set<Reservation> is_reserved_in_Reservation
    inverse Reservation::is_for_Room;
    attribute String RoomName;
    attribute Unsigned Short NrBeds;
    attribute Unsigned Short Cost;
  }
}

B&B Example – ODL for Reservation

interface Reservation {
  extent Reservations
  keys (RoomNr, ArrivalDate)
  : persistent {
    attribute Unsigned Short GuestNr;
    relationship Guest is_for_Guest
    inverse Guest::has_Reservation;
    attribute Unsigned Short RoomNr;
    relationship Room is_for_Room
    inverse Room::is_reserved_in_Reservation;
    attribute String ArrivalDate;
    attribute Unsigned Short NrDays;
  }
}
B&B Example – ODL for Guest

interface Guest {
    extent Guests
    keys GuestNr, (GuestName, StreetNr, City)
    ): persistent {
    attribute Unsigned Short GuestNr;
    relationship Reservation has_Reservation
        inverse Reservation::is_for_Guest;
    attribute String GuestName;
    attribute String StreetNr;
    attribute String City;
    }

OQL

• Basic SQL syntax: select-from-where
• additional flexibility
  – nesting: computed relations in select, from, and where clauses
  – expressions: path expressions, user-defined operators, use of collections such as array, list, and bag
OQL – Examples

List the name and address of Guests with reservations for more than one day.

```oql
select struct(x.GuestName, x.StreetNr, x.City)
from x in Guest, y in x.has_Reservation
where y.NrDays > 1
```

Is there a reservation for the Kennedy room on 13 May?

```oql
exists x in Reservation : x.ArrivalDate = "13 May"
   and x.is_for_Room.RoomName = "Kennedy"
```

For each room, list the cities and arrival dates of guests with reservations.

```oql
select struct(x.RoomName,
   (select struct(y.ArrivalDate, y.is_for_Guest.City
     from y in x.is_reserved_in_Reservation))
   from x in Room
```

ODMG C++

- ODMG design principle:
  - the programmer should see only one language (not one embedded in the other)
  - ODMG C++ should look like C++ (as much as possible)

- Problems
  - persistence – inherit from Persistent_Object and add Ref<> to provide access to the instances
  - relationships – extend the language with an inverse clause
  - programmer responsibility – enforce key constraints and other integrity constraints
**B&B Example – ODMG C++ Abstraction Diagram**

SiteOfInterest( View, Site )

 Room( RoomNr, RoomName, Cost, View,
    Reservation(ArrivalDate, NrDays, GuestNr )* )

Guest(GuestNr, GuestName, StreetNr, City )

**B&B Example – ODL C++ for SiteOfInterest and Guest Classes**

class Room; // forward declaration
class SiteOfInterest : public Persistent_Object {
    String View; // key
    Set< Ref<Reservation> > is_for_Room
        inverse Room::has_SiteOfInterest;
    String Site;
    static Ref< Set< Ref<SiteOfInterest> > > SitesOfInterest;
    static const char * const extent_name;
};
class Guest : public Persistent_Object {
    unsigned short GuestNr; // key
    String GuestName; // key (GuestName, StreetNr, City)
    String StreetNr;
    String City;
    static Ref< Set< Ref<Guest> > > Guests;
    static const char * const extent_name;
};
B&B Example – ODL C++ for Room Class

struct Reservation {
    Date ArrivalDate;
    unsigned short NrDays;
    unsigned short GuestNr;
    Ref<Guest> identifies_Guest;
};

class Room : public Persistent_Object {
    unsigned short RoomNr; // key
    String RoomName; // key
    unsigned short Cost;
    String View;
    Ref<SiteOfInterest> has_SiteOfInterest
        inverse SiteOfInterest::is_for_Room;
    Set<Reservation> Reservations;
    static Ref< Set< Ref<Room> > > Rooms;
    static const char * const extent_name;
};

B&B Example – OML C++ Service

void GetArrivingGuestList(const Date &today) {
    Transaction getArrivingGuestList;
    Set< Ref<Guest> > guests;
    getArrivingGuestList.begin();
    Guest::Guests = database->lookup_object(Guest::extent_name);
    cout << “Guests arriving on ” << today << “:” << endl;
    oql(guests,
        “select r.identifies_Guest \n        from r in (select x.Reservations from x in Room) \n        where r.ArrivalDate = $1”, today);
    ListArrivingGuests(guests);
    getArrivingGuestList.commit();
}
B&B Example – OML C++ Service (cont.)

```cpp
#include <iostream.h>
#include "schema.hxx"
static Database BandB_DB;
static void ListArrivingGuests(const Collection< Ref<Guest> > &guestSet) {
    Ref<Guest> guest;
    Iterator<Ref<Guest> > git = guestSet.create_iterator();
    while(git.next(guest))
        cout << guest->GuestName << ", " << guest->City << endl;
}
void GetArrivingGuestList(const Date &today) {
    ...
}
main() {
    BandB_DB.open("BandB");
    GetArrivingGuestList(Date.current());
    BandB_DB. close("BandB");
}
```

OSM Development Methodology

- Model-Driven Development
  - come to understand application
  - transform understanding through development into code
  - use theory and techniques
    - formalism – tunable, helps achieve better understanding
    - tool support
  - solve problems and achieve success
- Check Lists
  - process guide (not step by step, but ordered to help)
  - reminder about items that may be overlooked